

Astrostatistical Tools in Python

Tom Loredo, Alanna Connors, Travis Oliphant

Cornell/Eureka Scientific/BYU

Motivation

Many advanced methods are *conceptually* simple but *computationally* complex.

Competing methods of very different levels of sophistication are often similar from end-user's perspective.

Principal obstacle to understanding/use is the *art of statistical computing*.

Goal: Improve astrostatistical practice by eliminating this obstacle

Project Components

- The Inference package in Python
 - ▶ **Library:** Self-contained tools tailored to astronomers' needs; *multiple methods*
 - ▶ **Parametric Inference Engine (PIE):** Framework for parametric modeling; *multiple methodologies* (χ^2 , likelihood, Bayes) with unified interface
 - ▶ **SciPy:** Scientific computation support
- Development of new, alternative methodology
- Outreach

Outreach

Astrostat sessions at meetings; workshop organization

- "Making it Work: Principled 'Model Free Deconvolution' via Multiscale Methods" (AAS)
- "Current Challenges in Multi-Scale Analysis" (CfA Workshop)
- "Data Analysis Challenges in Solar Physics" (AAS)
- "Highly Structured Models for Spectral and Image Analysis in High Energy Astrophysics" (CMU Case Studies in Bayesian Statistics)
- PHYSTAT 2003, 2005
- "Astrostatistics Tools" (HEAD 2004)
- GRAVSTAT 2005

New Methodology

Bayesian alternatives to existing methods

- Hardness ratios for counting process data
- Kepler periodograms for exoplanet searching
- Survey analysis with source uncertainties (handling Eddington-Malmquist bias)
- Flexible spectral/image modeling with error estimates
- In progress: Bayesian EVM, log-Fourier models for arrival time data, spatio-temporal coincidence assessment, experimental design

The Inference Package: SciPy Support

— *Travis Oliphant* —

- Random number generators
 - ▶ 81 continuous & 11 discrete dist'ns
 - ▶ Evaluate dist'ns, moments, cumulants; estimators
- Optimization
 - ▶ MINPACK wrappers
 - ▶ Constrained optimization
- Numeric3 in progress (merges Numeric & STScI's numarray)
- ...

The Inference Package: Library

Tools implemented with self-contained functions and objects

- Tools for continuous data (signal + noise):
Regression w/ measurement error, Bayesian EVM,
Lomb-Scargle periodogram, Bretthorst algorithm,
Bayesian harmonic analysis, Kepler periodograms, ...
- Tools for discrete data (counts, points):
Intervals & limits for count data, period searching in
arrival time data (Rayleigh, Z_n^2 , GL, log-Fourier), ...

The Inference Package: PIE

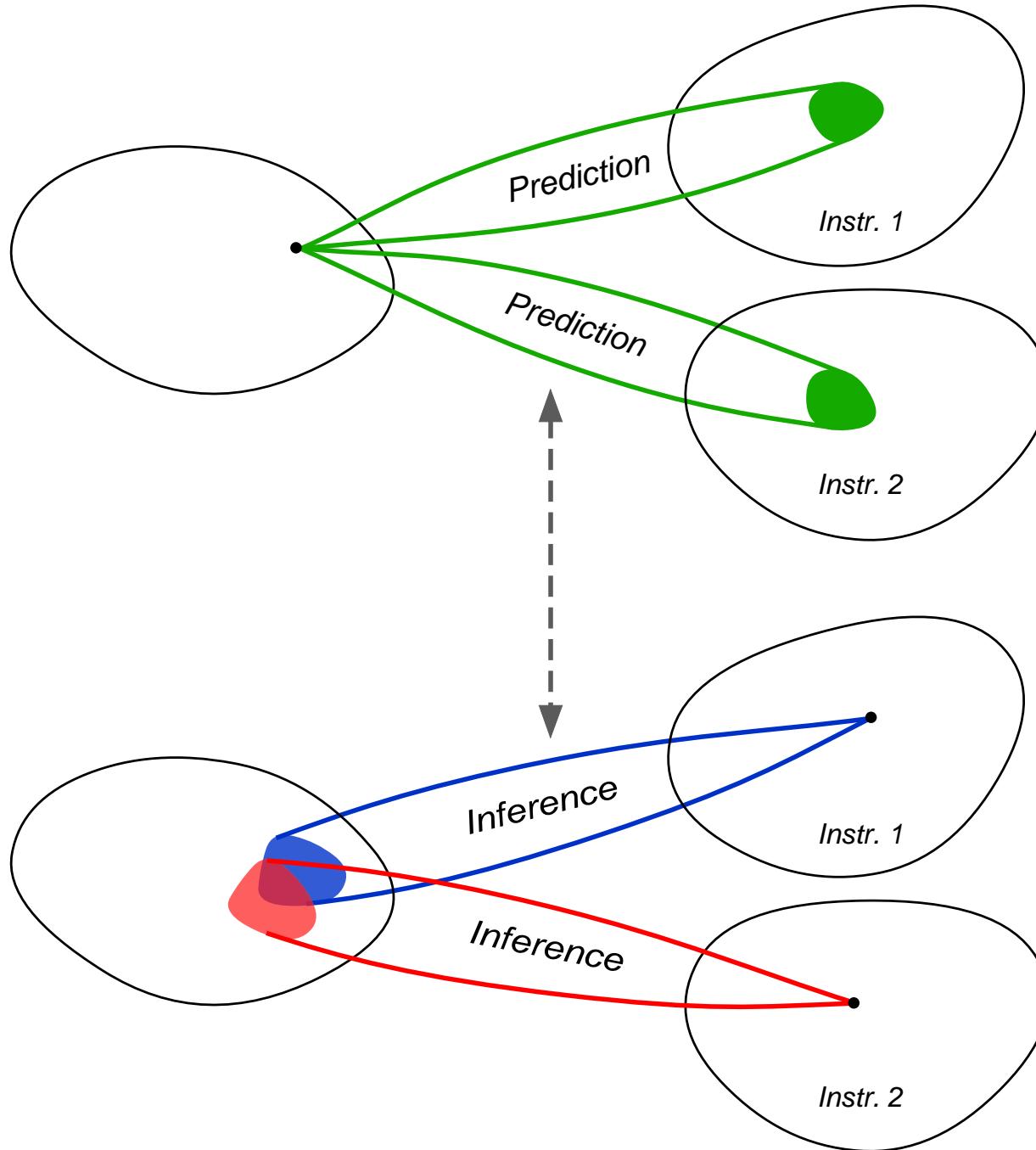
Parametric Inference Engine—A framework for inference with parametric models

- Three methodologies
 - ▶ Minimum χ^2
 - ▶ Maximum likelihood
 - ▶ Bayesian inference
- Multiple data types
 - ▶ Additive Gaussian noise (points, bins, response functions; on/off)
 - ▶ Poisson counting (as above)
 - ▶ Point process (exact and noisy)

- Automate standard parameter exploration tasks
 - ▶ Exploration on equispaced & logarithmic grids
 - ▶ Optimization (unconstrained/boundary constraints)
 - ▶ Exploration of subsets of parameter space (profiling/projection)
 - ▶ Information/covariance matrix calculation
- Bayesian computation
 - ▶ Marginalization and Bayes factors via quadrature & Laplace approximation
 - ▶ Calculation of 1-d, 2-d credible region boundaries
 - ▶ Basic Markov chain Monte Carlo (MCMC) support
- Simulate data

Parameter Space
Possible signals

Sample Space
Possible data



Inference With Three Mixin Classes

Collect the data as Predictors:

```
class MyData(PredictorSet):  
    d1 = SampledGaussianPred(data1, doc="Samples")  
    d2 = BinnedGaussianPred(data2, doc="Binned")
```

Build a model:

```
class PowerLaw(SignalModel):  
    A = PosParam(1., 'Amplitude')  
    alpha = RealParam(range=(-5,-1), 'Index')  
  
    def signal(self,E):  
        return self.A * E**self.alpha
```

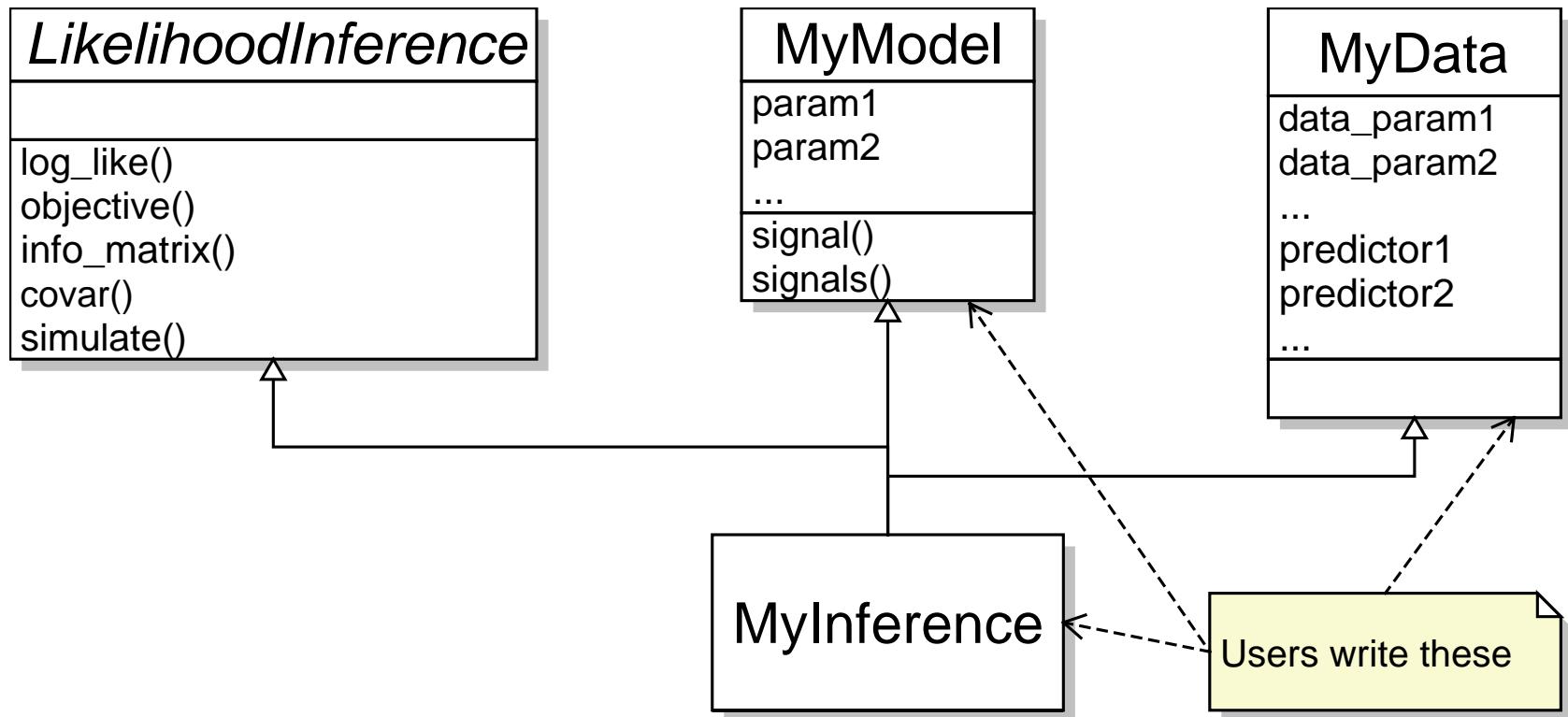
Combine in an inference:

```
class MyInf(LikelihoodInference, PowerLaw,  
MyData):  
    pass  
  
inf = MyInf()  
  
inf.A.logStep(1., 10., 50)  
inf.alpha.vary()  
  
grid = inf.fit()
```

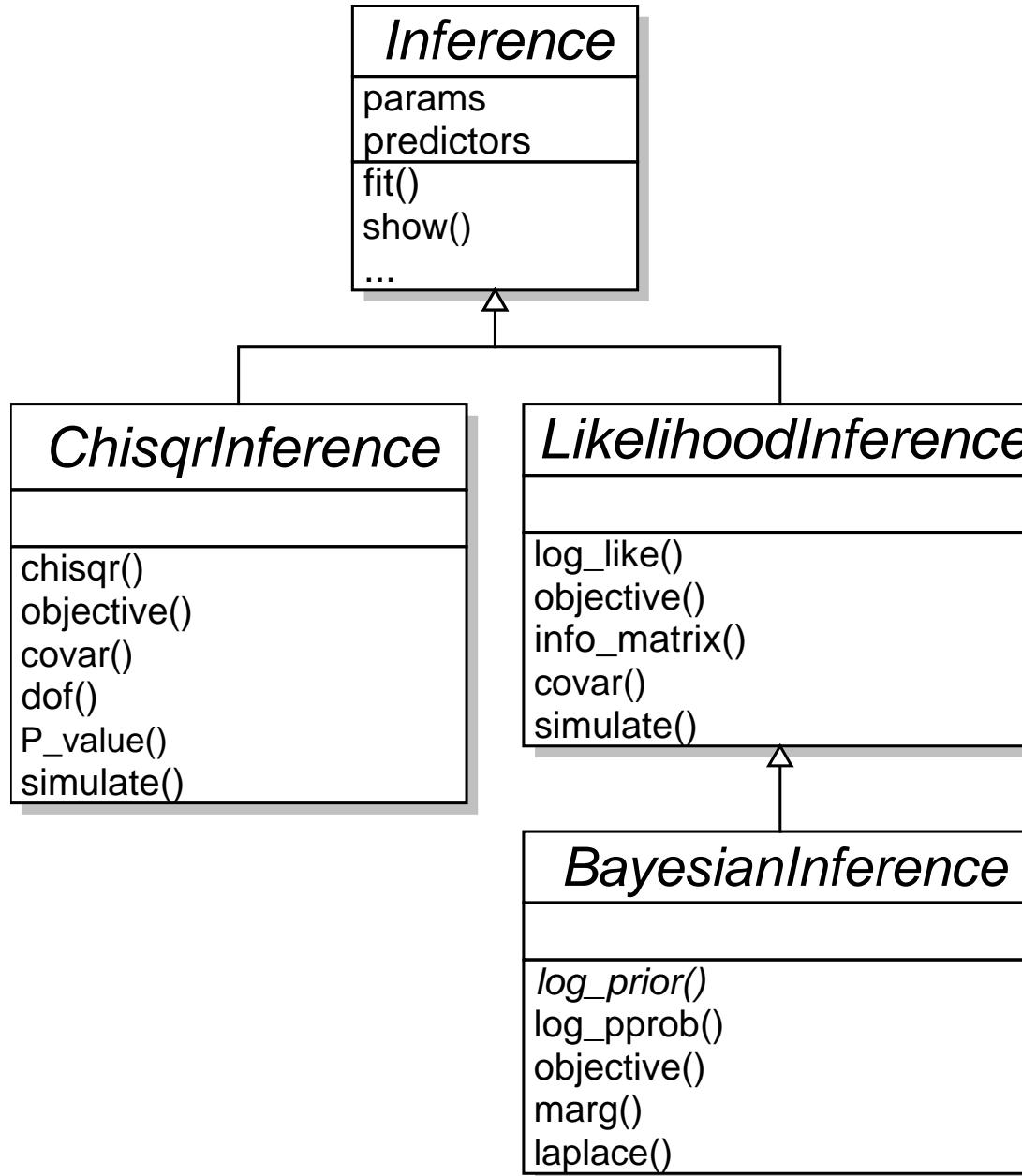
→ grid object contains profile likelihood, $\mathcal{L}_p(A)$

Implementation Highlights

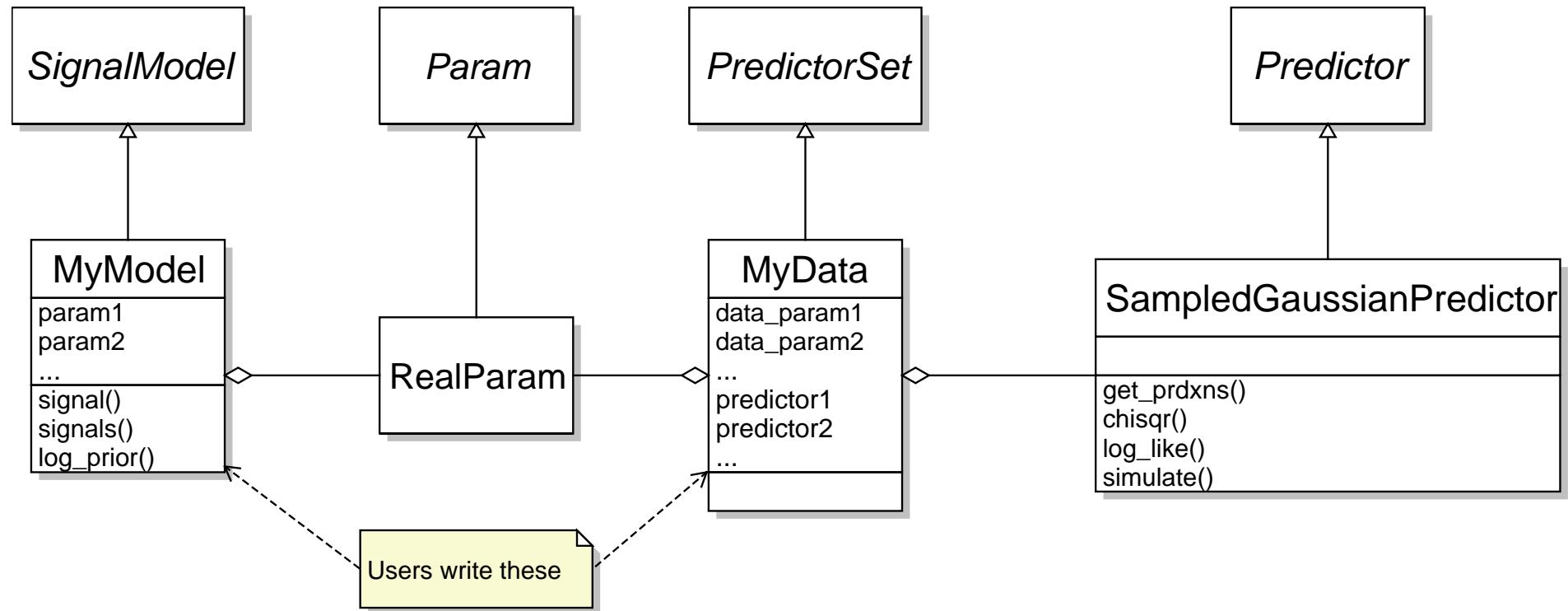
Top Level



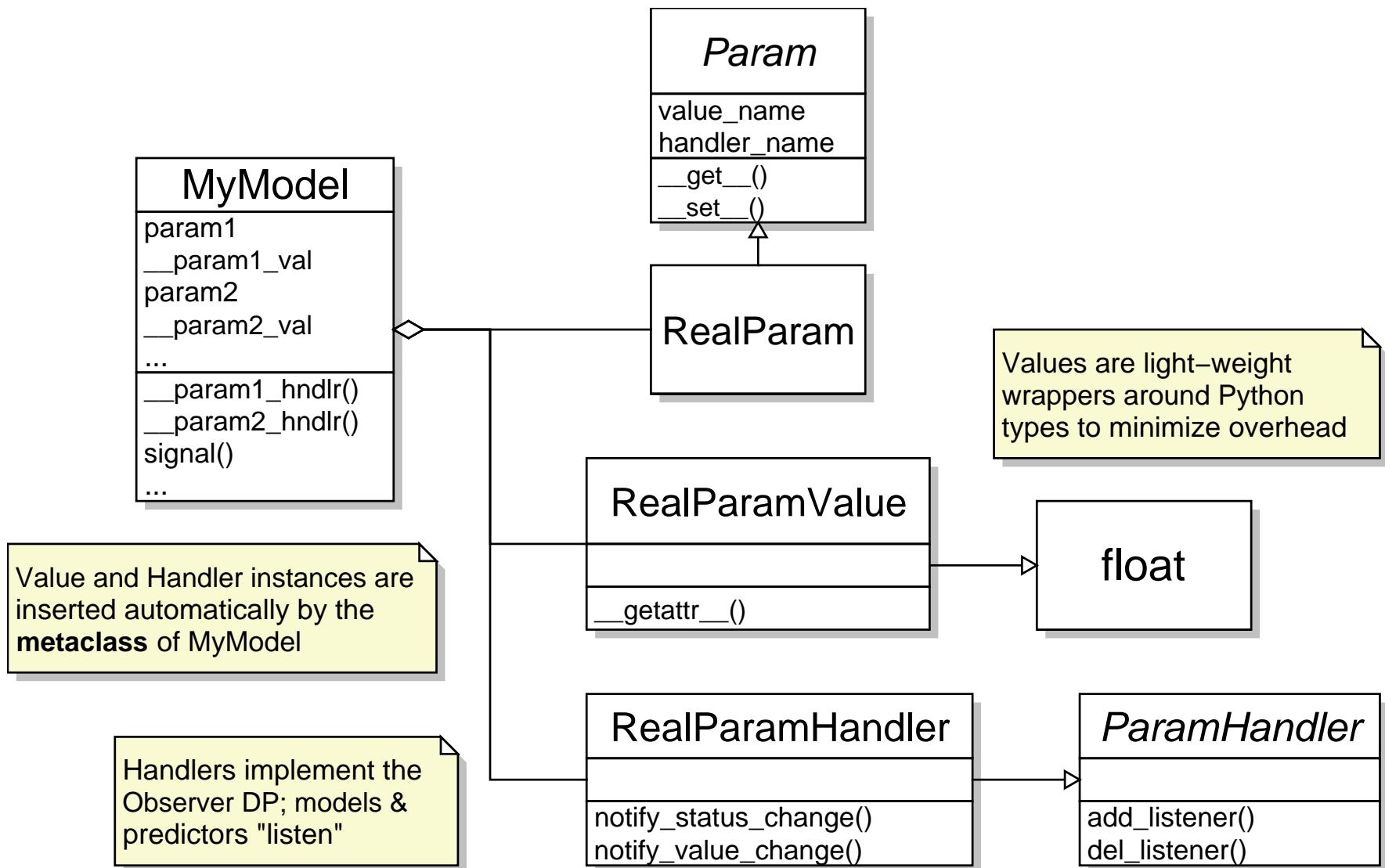
Inference Base Classes



Models & Data, Parameters & Predictors



Parameters via Descriptors & Metaclasses



Predictors have similar implementations.

Package release at SciPy.org imminent!

Thanks for your attention!